UNIT30: SERIALIZATION

【学習要項】
□Serialization
□Cereal

【演習手順】
1．Cereal を導入して skinned_mesh クラスにシリアリズ機能を実装する
①Cereal を GitHub からダウンロード(https://github.com/USCiLab/cereal)
※「Code」から「Download ZIP」を選択
②ダウンロードした ZIP を x3dgp.00 プロジェクトフォルダに展開
③3dgp プロパティページの「C/C++」→「全般」→「追加のインクルード…」に以下のパスを追加
.¥cereal-master¥include

2．skinned_mesh.h に必要なヘッダファイルをインクルードする

```
#include <cereal/archives/binary.hpp>
#include <cereal/types/memory.hpp>
#include <cereal/types/vector.hpp>
#include <cereal/types/set.hpp>
#include <cereal/types/unordered_map.hpp>
```

3．skinned_mesh.h に DirectXMath 構造体用の serialize テンプレート関数を定義する

```
 1:  namespace DirectX
 2:  {
 3:      template<class T>
 4:      void serialize(T& archive, DirectX::XMFLOAT2& v)
 5:      {
 6:          archive(
 7:              cereal::make_nvp("x", v.x),
 8:              cereal::make_nvp("y", v.y)
 9:          );
10:      }
11:
12:      template<class T>
13:      void serialize(T& archive, DirectX::XMFLOAT3& v)
14:      {
15:          archive(
16:              cereal::make_nvp("x", v.x),
17:              cereal::make_nvp("y", v.y),
18:              cereal::make_nvp("z", v.z)
19:          );
20:      }
21:
22:      template<class T>
23:      void serialize(T& archive, DirectX::XMFLOAT4& v)
24:      {
25:          archive(
26:              cereal::make_nvp("x", v.x),
27:              cereal::make_nvp("y", v.y),
28:              cereal::make_nvp("z", v.z),
29:              cereal::make_nvp("w", v.w)
30:          );
31:      }
32:
33:      template<class T>
34:      void serialize(T& archive, DirectX::XMFLOAT4X4& m)
35:      {
36:          archive(
37:              cereal::make_nvp("_11", m._11), cereal::make_nvp("_12", m._12),
38:              cereal::make_nvp("_13", m._13), cereal::make_nvp("_14", m._14),
39:              cereal::make_nvp("_21", m._21), cereal::make_nvp("_22", m._22),
40:              cereal::make_nvp("_23", m._23), cereal::make_nvp("_24", m._24),
41:              cereal::make_nvp("_31", m._31), cereal::make_nvp("_32", m._32),
42:              cereal::make_nvp("_33", m._33), cereal::make_nvp("_34", m._34),
43:              cereal::make_nvp("_41", m._41), cereal::make_nvp("_42", m._42),
```

```
44:             cereal::make_nvp("_43", m._43), cereal::make_nvp("_44", m._44)
45:         );
46:     }
47: }
```

4．シリアライズしたいデータを持つ構造体に serialize テンプレート関数を定義する
※メンバ変数・メンバ関数は省略してある

```
 1: struct skeleton
 2: {
 3:     struct bone
 4:     {
 5:         template<class T>
 6:         void serialize(T& archive)
 7:         {
 8:             archive(unique_id, name, parent_index, node_index, offset_transform);
 9:         }
10:     };
11:     template<class T>
12:     void serialize(T& archive)
13:     {
14:         archive(bones);
15:     }
16: };
17: struct animation
18: {
19:     struct keyframe
20:     {
21:         struct node
22:         {
23:             template<class T>
24:             void serialize(T& archive)
25:             {
26:                 archive(global_transform, scaling, rotation, translation);
27:             }
28:         };
29:         template<class T>
30:         void serialize(T& archive)
31:         {
32:             archive(nodes);
33:         }
34:     };
35:     template<class T>
36:     void serialize(T& archive)
37:     {
38:         archive(name, sampling_rate, sequence);
39:     }
40: };
41: struct scene
42: {
43:     struct node
44:     {
45:         template<class T>
46:         void serialize(T& archive)
47:         {
48:             archive(unique_id, name, attribute, parent_index);
49:         }
50:     };
51:     template<class T>
52:     void serialize(T& archive)
53:     {
54:         archive(nodes);
55:     }
56: };
57: class skinned_mesh
58: {
```

```
 59:  public:
 60:      struct vertex
 61:      {
 62:          template<class T>
 63:          void serialize(T& archive)
 64:          {
 65:              archive(position, normal, tangent, texcoord, bone_weights, bone_indices);
 66:          }
 67:      };
 68:      struct mesh
 69:      {
 70:          struct subset
 71:          {
 72:              template<class T>
 73:              void serialize(T& archive)
 74:              {
 75:                  archive(material_unique_id, material_name, start_index_location, index_count);
 76:              }
 77:          };
 78:          template<class T>
 79:          void serialize(T& archive)
 80:          {
 81:              archive(unique_id, name, node_index, subsets, default_global_transform,
 82:                  bind_pose, bounding_box, vertices, indices);
 83:          }
 84:      };
 85:      struct material
 86:      {
 87:          template<class T>
 88:          void serialize(T& archive)
 89:          {
 90:              archive(unique_id, name, Ka, Kd, Ks, texture_filenames);
 91:          }
 92:      };
 93:  };
```

5．skinned_mesh クラスのコンストラクタにコードを追加する

※このコードでは与えられた変数 fbx_filename の拡張子を“cereal”に変え、そのファイルが存在する場合はシリアライズされたデータからロードする。また、存在しない場合は従来通り FBX ファイルからロードする。

※skinnd_mesh クラスのデータ構造に変更があった場合はシリアライズされたファイルは無効になるので削除する必要がある。

```
  1:  skinned_mesh::skinned_mesh(ID3D11Device* device, const char* fbx_filename,
  2:      bool triangulate, float sampling_rate)
  3:  {
* 4:      std::filesystem::path cereal_filename(fbx_filename);
* 5:      cereal_filename.replace_extension("cereal");
* 6:      if (std::filesystem::exists(cereal_filename.c_str()))
* 7:      {
* 8:          std::ifstream ifs(cereal_filename.c_str(), std::ios::binary);
* 9:          cereal::BinaryInputArchive deserialization(ifs);
*10:          deserialization(scene_view, meshes, materials, animation_clips);
*11:      }
*12:      else
*13:      {
 14:
 15:          :
 16:          :
 17:          :
 18:
 19:          fbx_manager->Destroy();
 20:
*21:          std::ofstream ofs(cereal_filename.c_str(), std::ios::binary);
*22:          cereal::BinaryOutputArchive serialization(ofs);
*23:          serialization(scene_view, meshes, materials, animation_clips);
*24:      }
 25:      create_com_objects(device, fbx_filename);
```

```
26:  }
```

6．".¥¥resources¥¥nico.fbx"を skinned_mesh クラスのコンストラクタに指定し、1 回目と 2 回目のロード時間の違いを確認する

7．skinned_mesh クラスの append_animations メンバ関数でアニメーションを追加した場合はシリアライズの対象にならない
※設計を変更し上記問題に対応しなさい

【評価項目】
□シリアライゼーション